# Design Doc for CA 2

Zach Ahearn

K00282339

25/11/2023

A group of people in front of a city

Description automatically generated

Figure 1 - The start of the game

**INTRODUCTION**

My game is about a giant tentacle monster that attempts to destroy a city and its people. Unbeknownst to the monster, a wizard comes along to protect the people of the city. Using his magical shield barrier, he bounces the monster’s fireball attacks back against itself, holding out until the monster is defeated. The player is in control of the wizard’s shield barrier, moving it left and right with the arrow keys to intervene with the fireball before it reaches the citizens. The main focus of the game is to bounce the fireball back at the monster until its health bar is fully depleted and the boss is destroyed. My game is different because instead of the usual pong game of just counting your score every time you hit the ball, and losing lives when you miss the ball, my game works as a boss fight with health bars. The ball also increases in speed as the monster takes damage.

**GENERAL FEATURES**

**A computer screen shot of a program code

Description automatically generated**

Figure 2: Game starts when paused is false

The first feature of my game is its pause feature. The game starts paused and is only started once the enter key is pressed. This sets the paused variable to true and the game starts, as show in line 323 in figure 2. A similar variable is done to accept key inputs; relating to the arrow keys. The bat cannot be moved until the acceptInput variable is set to true, which only happens when the enter key is pressed.

A screen shot of a computer program

Description automatically generated

Figure 3: Example of sprites and textures

The second feature in my game is the use of many new sprites and textures. As seen in figure 3, I have many new sprites which affect the background and sets the scene for the game. This can be seen on lines 27 through 39 where I initialise the texture and sprite of the background. I load the file from its designated folder and set its position to 0, 0 which is the top left of the screen. Figure 1 shows how the game looks with all the sprites implemented.

A screen shot of a computer program

Description automatically generated

Figure 4: Code for player and boss health bars

The third feature in my game is the implementation of health bars for both the player and the boss. As seen in line 123 through 128 of figure 4, my player’s health bar is made up of a rectangle with a set width and height, a red color, and a position set near the bottom of the screen. My boss’s health bar is basically a copy of the player health bar except its position is set near the top of the screen. Again, figure 1 shows how these health bars look in game.

A screen shot of a computer program

Description automatically generated

Figure 5: Code for my sounds

My fourth game feature is that I added new sounds to the game. My sounds are straight forward and their variable make them clear what they are used for. As seen in figure 5, ‘playerDamage’ is when the player gets hurt, ‘bossDamage’ is when the boss is hurt, ‘bossMusicTrack’ is the music that plays throughout the game, and ‘shield’ is played every time the ball hits the bat.

A screen shot of a computer

Description automatically generated

Figure 6: Ball Speed Increase code

The fifth and final feature is my ball increasing in speed. Every time the ball touches the top of the screen, (damages the boss) this method is called is my ball.cpp file. On line 54 of figure 6, a float variable is made to equal the ball’s current speed (m\_Speed) and multiples by 0.06. Then in line 55, that variable is then added to the balls current speed. This essentially increases the balls speed by 6 percent each time.

**GAME PLAY**

**A screen shot of a computer program

Description automatically generated**

Figure 7: Handling the wizard's animation

In my game, I have a wizard sprite that as three different textures. In order to add animations to my wizard, I have its animation change based on a counter that increases ever time the ball intersects with the bat. This can be seen in figure 7 between lines 369 and 375. Then in lines 377 through 393, I have multiple if-else statements to handle the animation change. If the counter is equal to 1, then it sets the wizard sprite’s texture to the first texture. This is the same for animations 2 and 3. In order to loop the animation, if the counter is equal to 4, the counter is set back to 1.

A computer screen shot of a program

Description automatically generated

Figure 8: Implementing reduction in health bars

In figure 8, lines 336 through 345 handle the reduction of the player’s health bar every time the ball reaches the bottom of the screen. The width of the health bar is decreased by 200 pixels and the size is set every time. In line 352, the boss’s health bar is handled the same way when the ball hits the top of the screen but only decreases by 95 pixels. Essentially, the player can take 10 hits and the boss takes 20.

A screen shot of a computer program

Description automatically generated

Figure 9: Overlaying sprites

The final implementation I want to address is how I overlayed sprites on the ball and bat. I made new variables which are equal to the x and y position of the ball and bat. The position of the sprites are then set to the position of these new variable and a set number is taken away from the variables for each sprite so they overlay in the correct positions.

**CONCLUSION**

I found this project to not be too difficult, especially after having learned so much from my previous project with SFML. The only real issue I experienced was how to implement the sprites to overlay the ball and bat. I initially did not know that ball.getPosition().left was referring to the x position of the ball and ball.getPosition().top was referring to its y position. Setting the position of a sprite to just ball.getPosition() did not work and I learned how to fix this.